**实验项目五 基于Gazebo的机器人仿真**

1. **实验目的**

1、 掌握URDF文件的编程；

2、 掌握Gazebo软件的使用。

1. **实验设备**

1、 雷达模块一台；

2、 无线键鼠一套；

3、 显示器一套。

1. **实验步骤和内容**

本节主要介绍URDF和Gazebo的基本集成流程；要在Gazebo中显示机器人，URDF需要进行额外的配置；以及关于Gazebo仿真环境的搭建。

### 1、URDF与Gazebo基本集成流程

**（1）创建功能包，导入依赖项；**

创建新功能包，导入依赖包：urdf、xacro、gazebo\_ros、gazebo\_ros\_control、gazebo\_plugins。

**（2）编写URDF或者Xacro文件；**

创建一个机器人模型（盒状即可），显示在Gazebo中：

<robot name="mycar">

<link name="base\_link">

<visual>

<geometry>

<box size="0.5 0.3 0.1" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

<material name="yellow">

<color rgba="0.5 0.3 0.0 1" />

</material>

</visual>

<collision>

<geometry>

<box size="0.5 0.3 0.1" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

</collision>

<inertial>

<origin xyz="0 0 0" />

<mass value="2" />

<inertia ixx="1" ixy="0" ixz="0" iyy="0" iyz="1" izz="1" />

</inertial>

</link>

<gazebo reference="base\_link">

<material>Gazebo/Red</material>

</gazebo>

</robot>

**注意：**当URDF需要与Gazebo集成时，和Rviz有明显区别：

1. 必须使用collision标签，因为既然是仿真环境，那么必然涉及到碰撞检测，collision提供碰撞检测的依据。
2. 必须使用inertial标签，此标签标注了当前机器人某个刚体部分的惯性矩阵，用于一些力学相关的仿真计算。
3. 颜色设置，也需要重新使用gazebo标签标注，因为之前的颜色设置为了方便调试包含透明度，仿真环境下没有此选项。

**（3）启动Gazebo并显示机器人模型**

launch文件实现：

<launch>

<!-- 将 Urdf 文件的内容加载到参数服务器 -->

<param name="robot\_description" textfile="$(find test\_03)/urdf/car.urdf" />

<!-- 启动 gazebo 仿真环境，当前环境为空环境 -->

<include file="$(find gazebo\_ros)/launch/empty\_world.launch" />

<!-- 在 gazebo 中显示机器人模型 -->

<node pkg="gazebo\_ros" type="spawn\_model" name="model" args="-urdf -model mycar -param robot\_description" />

<!--

在 Gazebo 中加载一个机器人模型，该功能由 gazebo\_ros 下的 spawn\_model 提供:

-urdf 加载的是 urdf 文件

-model mycar 模型名称是 mycar

-param robot\_description 从参数 robot\_description 中载入模型

-x 模型载入的 x 坐标

-y 模型载入的 y 坐标

-z 模型载入的 z 坐标

-->

</launch>

**2、URDF集成Gazebo相关设置**

Gazebo在集成URDF时，需要做一些修改。比如：必须添加collision碰撞属性相关参数；必须添加inertial惯性矩阵相关参数；除此之外，如果移植Rviz中机器人的颜色设置是没有显示的，颜色设置也必须要做出相应的变更。

**（1）collision**

如果机器人link是标准的几何体形状，和link的visual属性设置一致即可。

**（2）inertial**

惯性矩阵的设置需要结合link的质量与外形参数动态生成，标准的球体、圆柱与立方体的惯性矩阵公式如下（已经封装为xacro实现）。

1）球体惯性矩阵：

<!-- Macro for inertia matrix -->

<xacro:macro name="sphere\_inertial\_matrix" params="m r">

<inertial>

<mass value="${m}" />

<inertia ixx="${2\*m\*r\*r/5}" ixy="0" ixz="0"

iyy="${2\*m\*r\*r/5}" iyz="0"

izz="${2\*m\*r\*r/5}" />

</inertial>

</xacro:macro>

2）圆柱惯性矩阵：

<xacro:macro name="cylinder\_inertial\_matrix" params="m r h">

<inertial>

<mass value="${m}" />

<inertia ixx="${m\*(3\*r\*r+h\*h)/12}" ixy = "0" ixz = "0"

iyy="${m\*(3\*r\*r+h\*h)/12}" iyz = "0"

izz="${m\*r\*r/2}" />

</inertial>

</xacro:macro>

3）立方体惯性矩阵：

<xacro:macro name="Box\_inertial\_matrix" params="m l w h">

<inertial>

<mass value="${m}" />

<inertia ixx="${m\*(h\*h + l\*l)/12}" ixy = "0" ixz = "0"

iyy="${m\*(w\*w + l\*l)/12}" iyz= "0"

izz="${m\*(w\*w + h\*h)/12}" />

</inertial>

</xacro:macro>

需要注意的是，原则上，除了base\_footprint外，机器人的每个刚体部分都需要设置惯性矩阵，且惯性矩阵必须经计算得出，如果随意定义刚体部分的惯性矩阵，那么可能会导致机器人在Gazebo中出现抖动，移动等现象。

**（3）颜色设置**

在gazebo中显示link的颜色，必须要使用指定的标签：

<gazebo reference="link节点名称">

<material>Gazebo/Blue</material>

</gazebo>

**注意：**material标签中，设置的值区分大小写，颜色可以设置为Red Blue Green Black ...

1. **Gazebo仿真环境搭建**

Gazebo中创建仿真实现方式有两种：

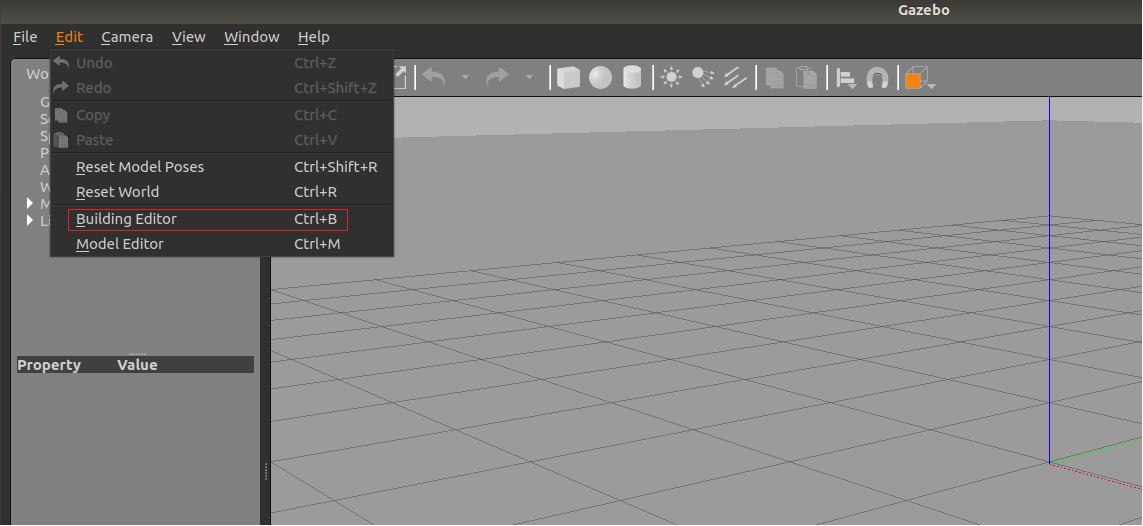
方式1：直接添加内置组件创建仿真环境

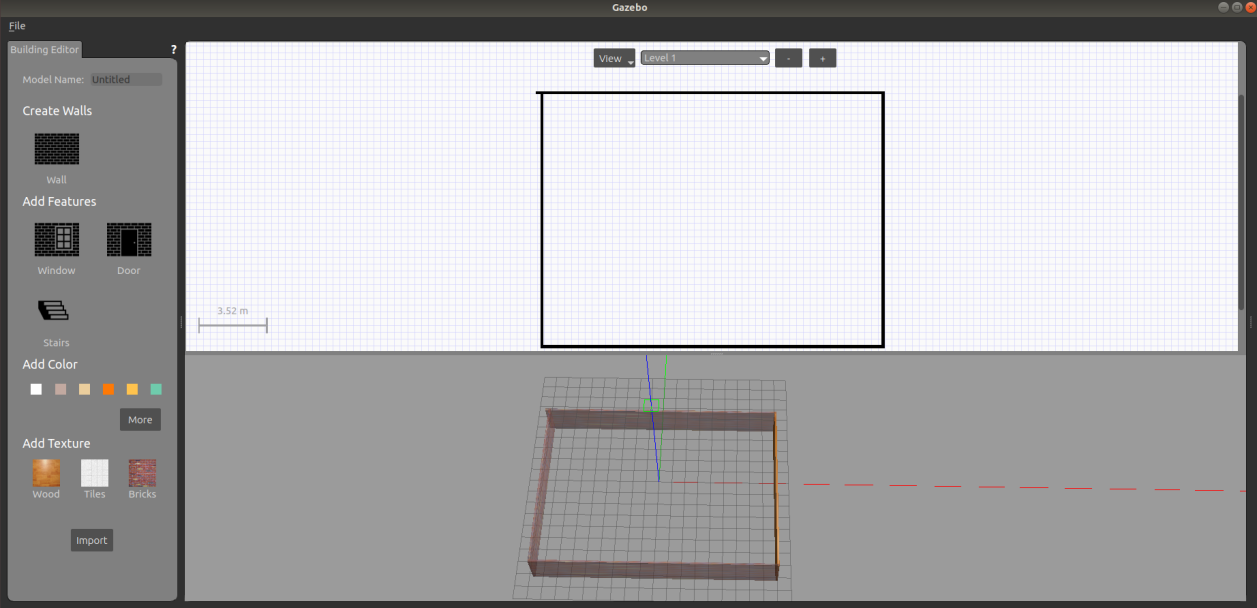
方式2：手动绘制仿真环境（更为灵活）

也还可以直接下载使用官方或第三方提高的仿真环境插件。

**（1）自定义仿真环境**

1）启动gazebo打开构建面板，绘制仿真环境





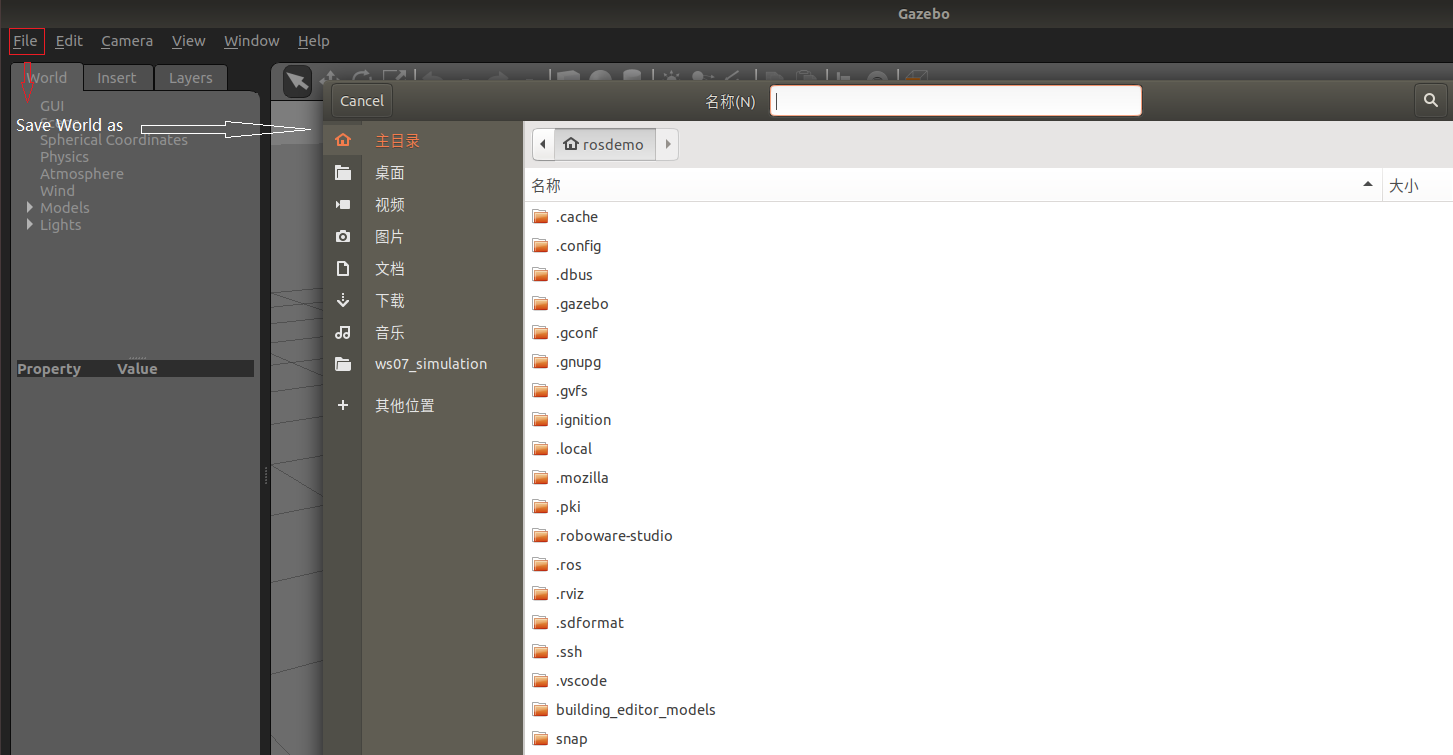
2）保存构建的环境

点击：左上角file ---> Save（保存路径功能包下的：models）

然后file ---> Exit Building Editor

3）保存为world文件

保存为world文件（保存路径功能包下的：worlds）。



4）启动

下面的launch文件中添加的URDF文件是前期我们创造的机器人模型，启动gazebo仿真环境中，如果想启动自己刚才创建的.world文件，需要确保路径和名字正确。此launch文件中是创建好的提供参考的一个world文件，可直接运行。

<launch>

<!-- 将 Urdf 文件的内容加载到参数服务器 -->

<param name="robot\_description" command="$(find xacro)/xacro $(find test\_03)/xacro/mycar.xacro" />

<!-- 启动 gazebo 仿真环境 -->

<include file="$(find gazebo\_ros)/launch/empty\_world.launch">

<arg name="world\_name" value="$(find test\_03)/worlds/box\_house.world" />

</include>

<!-- 在 gazebo 中显示机器人模型 -->

<node pkg="gazebo\_ros" type="spawn\_model" name="model" args="-urdf -model mycar -param robot\_description" />

</launch>

**（2）使用官方提供的插件**

当前Gazebo提供的仿真道具有限，还可以下载官方支持，可以提供更为丰富的仿真实现，具体实现如下：

1）下载官方模型库

$ git clone https://github.com/osrf/gazebo\_models

**注意：**此过程可能比较耗时。

2）将模型库复制进gazebo

将得到的gazebo\_models文件夹内容复制到/usr/share/gazebo-\*/models

3）应用

重启Gazebo，选择左侧菜单栏的insert可以选择并插入相关道具了

1. **实验参考结果**

将之前的机器人模型（xacro版）显示在gazebo中

1. **编写封装惯性矩阵算法的xacro文件**

查看test\_03功能包下，xacro文件夹里的head.xacro文件。

<robot name="base" xmlns:xacro="http://wiki.ros.org/xacro">

<xacro:macro name="sphere\_inertial\_matrix" params="m r">

<inertial>

<mass value="${m}" />

<inertia ixx="${2\*m\*r\*r/5}" ixy="0" ixz="0"

iyy="${2\*m\*r\*r/5}" iyz="0"

izz="${2\*m\*r\*r/5}" />

</inertial>

</xacro:macro>

<xacro:macro name="cylinder\_inertial\_matrix" params="m r h">

<inertial>

<mass value="${m}" />

<inertia ixx="${m\*(3\*r\*r+h\*h)/12}" ixy = "0" ixz = "0"

iyy="${m\*(3\*r\*r+h\*h)/12}" iyz = "0"

izz="${m\*r\*r/2}" />

</inertial>

</xacro:macro>

<xacro:macro name="Box\_inertial\_matrix" params="m l w h">

<inertial>

<mass value="${m}" />

<inertia ixx="${m\*(h\*h + l\*l)/12}" ixy = "0" ixz = "0"

iyy="${m\*(w\*w + l\*l)/12}" iyz= "0"

izz="${m\*(w\*w + h\*h)/12}" />

</inertial>

</xacro:macro>

</robot>

1. **复制相关xacro文件，并设置collision,inertial和color等参数**

（1）底盘Xacro文件（xacro文件夹里car\_base.xacro文件）

<robot name="car\_base" xmlns:xacro="http://www.ros.org/wiki/xacro">

<xacro:property name="PI" value="3.1415927"/>

<material name="black">

<color rgba="0.0 0.0 0.0 1.0" />

</material>

<xacro:property name="base\_footprint\_radius" value="0.001" />

<xacro:property name="base\_link\_radius" value="0.1" />

<xacro:property name="base\_link\_length" value="0.08" />

<xacro:property name="earth\_space" value="0.015" />

<xacro:property name="base\_link\_m" value="0.5" />

<link name="base\_footprint">

<visual>

<geometry>

<sphere radius="${base\_footprint\_radius}" />

</geometry>

</visual>

</link>

<link name="base\_link">

<visual>

<geometry>

<cylinder radius="${base\_link\_radius}" length="${base\_link\_length}" />

</geometry>

<origin xyz="0 0 0" rpy="0 0 0" />

<material name="yellow">

<color rgba="0.8 0.3 0.1 1.0" />

</material>

</visual>

<collision>

<geometry>

<cylinder radius="${base\_link\_radius}" length="${base\_link\_length}" />

</geometry>

<origin xyz="0 0 0" rpy="0 0 0" />

</collision>

<xacro:cylinder\_inertial\_matrix m="${base\_link\_m}" r="${base\_link\_radius}" h="${base\_link\_length}" />

</link>

<gazebo reference="base\_link">

<material>Gazebo/Orange</material>

</gazebo>

<joint name="base\_link\_to\_base\_footprint" type="fixed">

<parent link="base\_footprint" />

<child link="base\_link" />

<origin xyz="0 0 ${earth\_space + base\_link\_length / 2 }" />

</joint>

<xacro:property name="wheel\_radius" value="0.0325" />

<xacro:property name="wheel\_length" value="0.015" />

<xacro:property name="wheel\_m" value="0.05" />

<xacro:macro name="add\_wheels" params="name flag">

<link name="${name}\_wheel">

<visual>

<geometry>

<cylinder radius="${wheel\_radius}" length="${wheel\_length}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="${PI / 2} 0.0 0.0" />

<material name="black" />

</visual>

<collision>

<geometry>

<cylinder radius="${wheel\_radius}" length="${wheel\_length}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="${PI / 2} 0.0 0.0" />

</collision>

<xacro:cylinder\_inertial\_matrix m="${wheel\_m}" r="${wheel\_radius}" h="${wheel\_length}" />

</link>

<gazebo reference="${name}\_wheel">

<material>Gazebo/Red</material>

</gazebo>

<joint name="${name}\_wheel\_to\_base\_link" type="continuous">

<parent link="base\_link" />

<child link="${name}\_wheel" />

<origin xyz="0 ${flag \* base\_link\_radius} ${-(earth\_space + base\_link\_length / 2 - wheel\_radius) }" />

<axis xyz="0 1 0" />

</joint>

</xacro:macro>

<xacro:add\_wheels name="left" flag="1" />

<xacro:add\_wheels name="right" flag="-1" />

<xacro:property name="support\_wheel\_radius" value="0.0075" />

<xacro:property name="support\_wheel\_m" value="0.03" />

<xacro:macro name="add\_support\_wheel" params="name flag" >

<link name="${name}\_wheel">

<visual>

<geometry>

<sphere radius="${support\_wheel\_radius}" />

</geometry>

<origin xyz="0 0 0" rpy="0 0 0" />

<material name="black" />

</visual>

<collision>

<geometry>

<sphere radius="${support\_wheel\_radius}" />

</geometry>

<origin xyz="0 0 0" rpy="0 0 0" />

</collision>

<xacro:sphere\_inertial\_matrix m="${support\_wheel\_m}" r="${support\_wheel\_radius}" />

</link>

<gazebo reference="${name}\_wheel">

<material>Gazebo/Red</material>

</gazebo>

<joint name="${name}\_wheel\_to\_base\_link" type="continuous">

<parent link="base\_link" />

<child link="${name}\_wheel" />

<origin xyz="${flag \* (base\_link\_radius - support\_wheel\_radius)} 0 ${-(base\_link\_length / 2 + earth\_space / 2)}" />

<axis xyz="1 1 1" />

</joint>

</xacro:macro>

<xacro:add\_support\_wheel name="front" flag="1" />

<xacro:add\_support\_wheel name="back" flag="-1" />

</robot>

**注意：**如果机器人模型在Gazebo中产生了抖动，滑动，缓慢位移...诸如此类情况，请查看：

* 惯性矩阵是否设置了，且设置是否正确合理；
* 车轮翻转需要依赖于PI值，如果PI值精度偏低，也可能导致上述情况产生。

（2）摄像头Xacro文件（xacro文件夹里camera.xacro文件）

<robot name="my\_camera" xmlns:xacro="http://wiki.ros.org/xacro">

<xacro:property name="camera\_length" value="0.01" />

<xacro:property name="camera\_width" value="0.025" />

<xacro:property name="camera\_height" value="0.025" />

<xacro:property name="camera\_x" value="0.08" />

<xacro:property name="camera\_y" value="0.0" />

<xacro:property name="camera\_z" value="${base\_link\_length / 2 + camera\_height / 2}" />

<xacro:property name="camera\_m" value="0.01" />

<link name="camera">

<visual>

<geometry>

<box size="${camera\_length} ${camera\_width} ${camera\_height}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

<material name="black" />

</visual>

<collision>

<geometry>

<box size="${camera\_length} ${camera\_width} ${camera\_height}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

</collision>

<xacro:Box\_inertial\_matrix m="${camera\_m}" l="${camera\_length}" w="${camera\_width}" h="${camera\_height}" />

</link>

<gazebo reference="camera">

<material>Gazebo/Blue</material>

</gazebo>

<joint name="camera\_to\_base\_link" type="fixed">

<parent link="base\_link" />

<child link="camera" />

<origin xyz="${camera\_x} ${camera\_y} ${camera\_z}" />

</joint>

</robot>

（3）雷达Xacro文件（laser.xacro文件）

<robot name="my\_laser" xmlns:xacro="http://wiki.ros.org/xacro">

<xacro:property name="support\_length" value="0.15" />

<xacro:property name="support\_radius" value="0.01" />

<xacro:property name="support\_x" value="0.0" />

<xacro:property name="support\_y" value="0.0" />

<xacro:property name="support\_z" value="${base\_link\_length / 2 + support\_length / 2}" />

<xacro:property name="support\_m" value="0.02" />

<link name="support">

<visual>

<geometry>

<cylinder radius="${support\_radius}" length="${support\_length}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

<material name="red">

<color rgba="0.8 0.2 0.0 0.8" />

</material>

</visual>

<collision>

<geometry>

<cylinder radius="${support\_radius}" length="${support\_length}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

</collision>

<xacro:cylinder\_inertial\_matrix m="${support\_m}" r="${support\_radius}" h="${support\_length}" />

</link>

<gazebo reference="support">

<material>Gazebo/White</material>

</gazebo>

<joint name="support\_to\_base\_link" type="fixed">

<parent link="base\_link" />

<child link="support" />

<origin xyz="${support\_x} ${support\_y} ${support\_z}" />

</joint>

<xacro:property name="laser\_length" value="0.05" />

<xacro:property name="laser\_radius" value="0.03" />

<xacro:property name="laser\_x" value="0.0" />

<xacro:property name="laser\_y" value="0.0" />

<xacro:property name="laser\_z" value="${support\_length / 2 + laser\_length / 2}" />

<xacro:property name="laser\_m" value="0.1" />

<link name="laser">

<visual>

<geometry>

<cylinder radius="${laser\_radius}" length="${laser\_length}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

<material name="black" />

</visual>

<collision>

<geometry>

<cylinder radius="${laser\_radius}" length="${laser\_length}" />

</geometry>

<origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />

</collision>

<xacro:cylinder\_inertial\_matrix m="${laser\_m}" r="${laser\_radius}" h="${laser\_length}" />

</link>

<gazebo reference="laser">

<material>Gazebo/Black</material>

</gazebo>

<joint name="laser\_to\_support" type="fixed">

<parent link="support" />

<child link="laser" />

<origin xyz="${laser\_x} ${laser\_y} ${laser\_z}" />

</joint>

</robot>

（4）组合底盘、摄像头与雷达的Xacro文件（在xacro文件夹里自己创建）

<robot name="mycar" xmlns:xacro="http://wiki.ros.org/xacro">

<xacro:include filename="head.xacro" />

<xacro:include filename="car\_base.xacro" />

<xacro:include filename="camera.xacro" />

<xacro:include filename="laser.xacro" />

</robot>

1. **在launch文件中启动gazebo并添加机器人模型（在launch文件夹新建）。**

**注意标红位置，需更改为上文自己新建xacro文件的名字。**

<launch>

<!-- 将 Urdf 文件的内容加载到参数服务器 -->

<param name="robot\_description" command="$(find xacro)/xacro $(find test\_03)/xacro/mycar.xacro" />

<!-- 启动 gazebo 仿真环境 -->

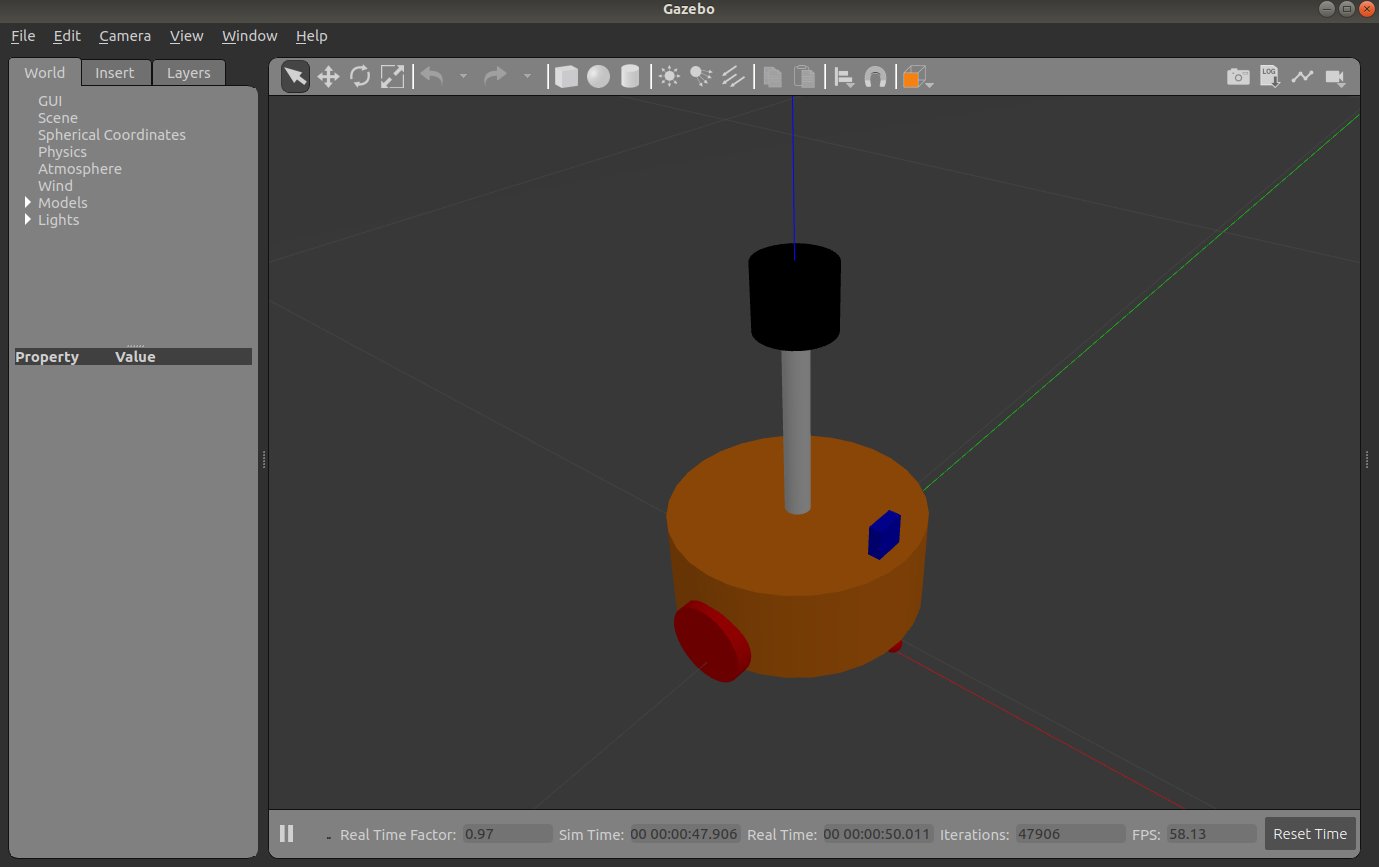
<include file="$(find gazebo\_ros)/launch/empty\_world.launch" />

<!-- 在 gazebo 中显示机器人模型 -->

<node pkg="gazebo\_ros" type="spawn\_model" name="model" args="-urdf -model mycar -param robot\_description" />

</launch>

**结果演示如下图：**



1. **相关的测试代码**

见代码包test\_03。